Augmenting trading systems with Hidden Markov Models

Luis Damiano
Department of Statistics, Iowa State University

Michael Weylandt & Brian Peterson

May 17, 2019

Goal

Illustrate how to use a Hidden Markov Model to extract a sequence of latent states from a series of observeations.

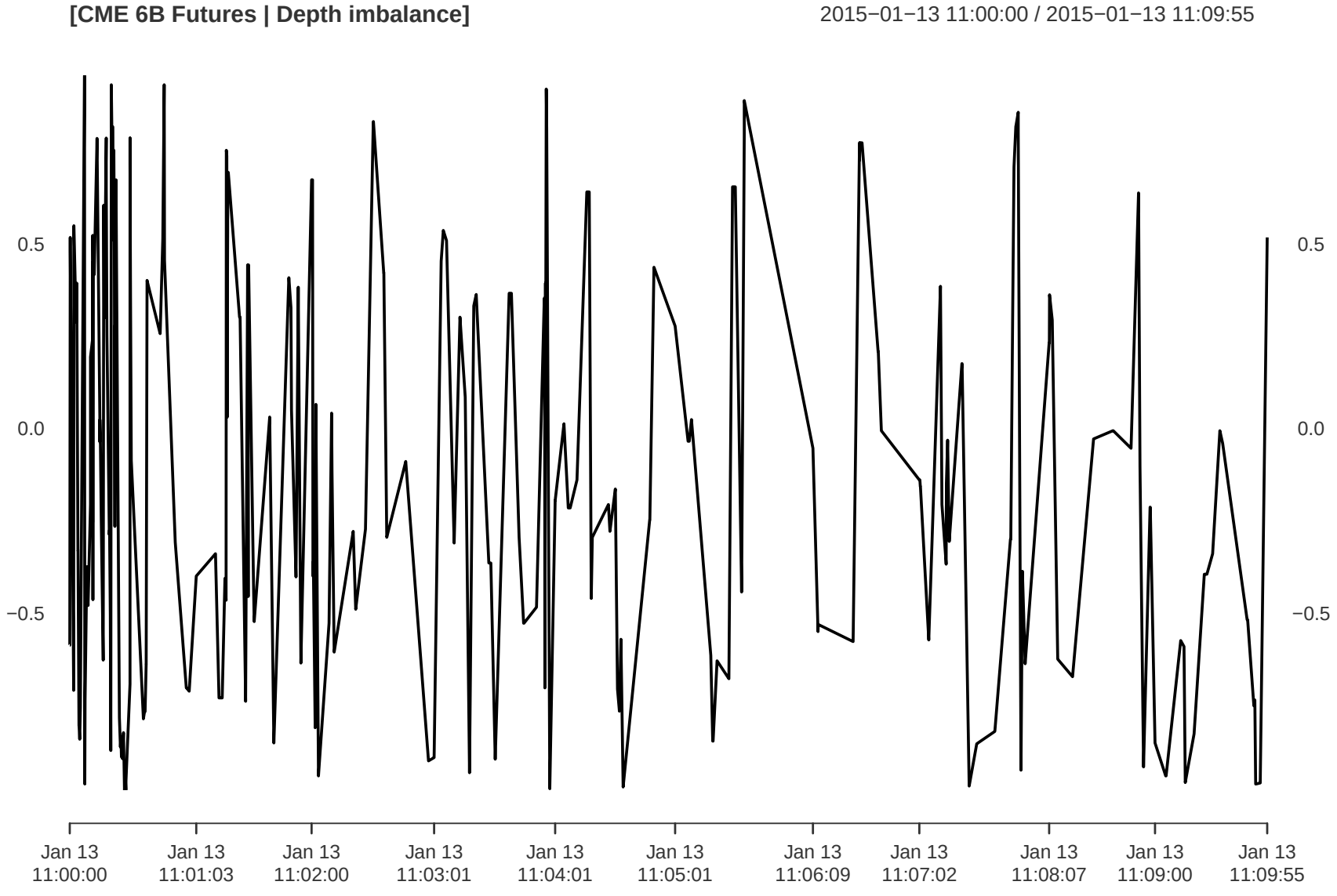
- Toy example.
- Key methodological aspects.
- Keep it intuitive (references on the back, details during the break).

Main idea

- We would like to incorporate new information in our system.
- Problem: the time series is too noisy to be fed directly to our decision models.

Motivating example

Challenge: describe the series without using the term **ugly**.



Motivating example

My attempt:

- Not *iid* but no clear time structure either: what model?
- No apparent trend: where is the signal?
- Too low signal-to-noise ratio to be used directly as input.
- Large dataset (~6k observations in 10 minutes).
- Unequally spaced observations.
  Overall, the series structure is ~~ugly~~ complex.

Latent variables

***Twist**: do not use the observed series directly.*

Where is the signal?

- We believe that there is a signal but it is **hidden**.
- Sometimes, the signal is not the observation itself but some **characteristics of the observation**.
  - E.g. it is not the value of the return itself, but its variability.
  - The series distribution takes different *configurations* over time.
  - What is the configuration that would best explain the observed value at a given time step?

Signal extraction

*Create **discrete features** based on latent states.*

E.g. from L1 data to...

- **Trader**: bullish/bearish.
- **Risk analyst**: low/high volatility.
- **Behavioural finance**: risk on/off.
- **Macroeconomist**: expansion/recession.

Features

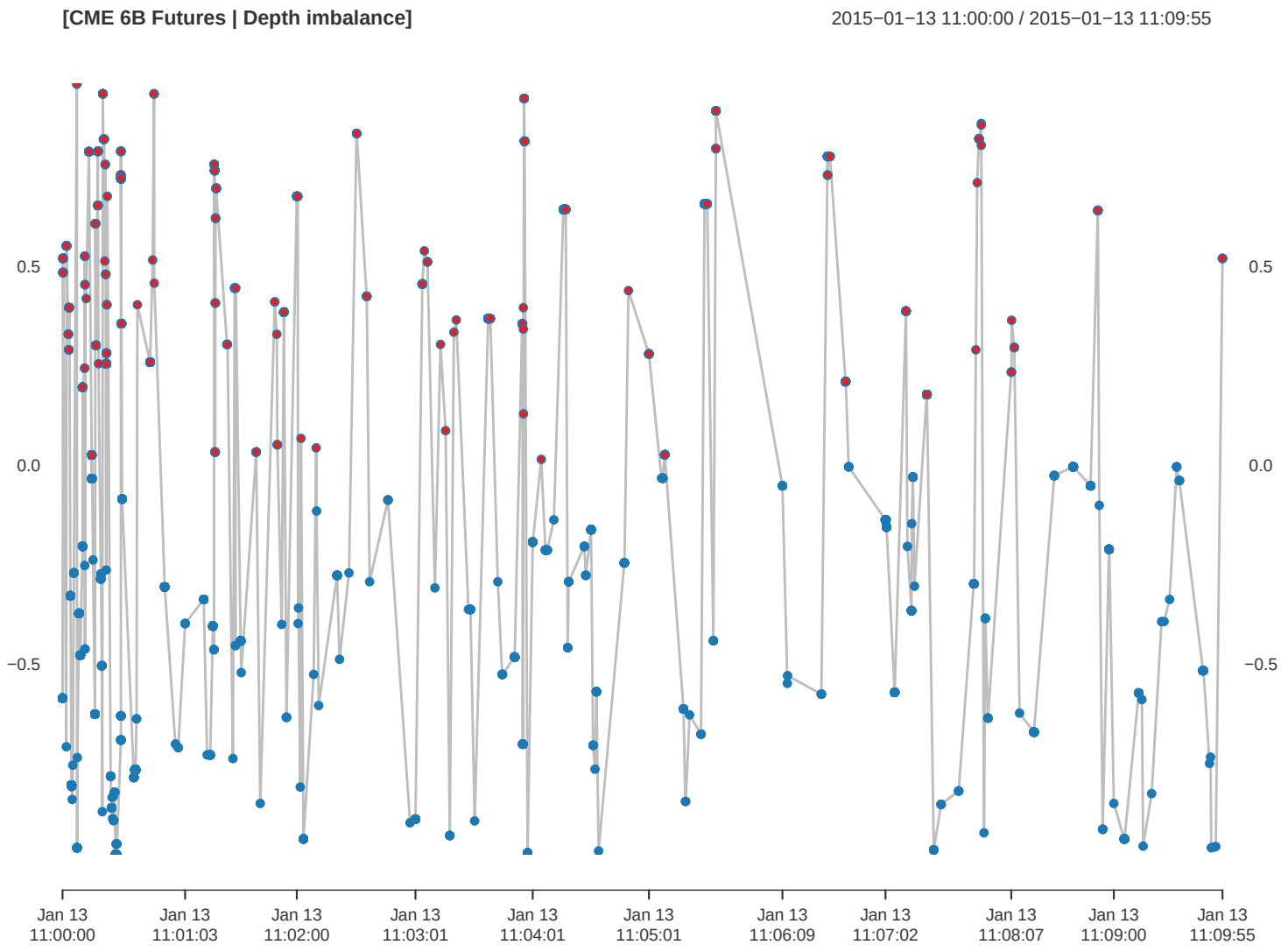
> *[. . .] some machine learning projects succeed and somefail. What makes the difference? Easily the most important factor is the features used. (Domingos 2012)*

What would make features strong?

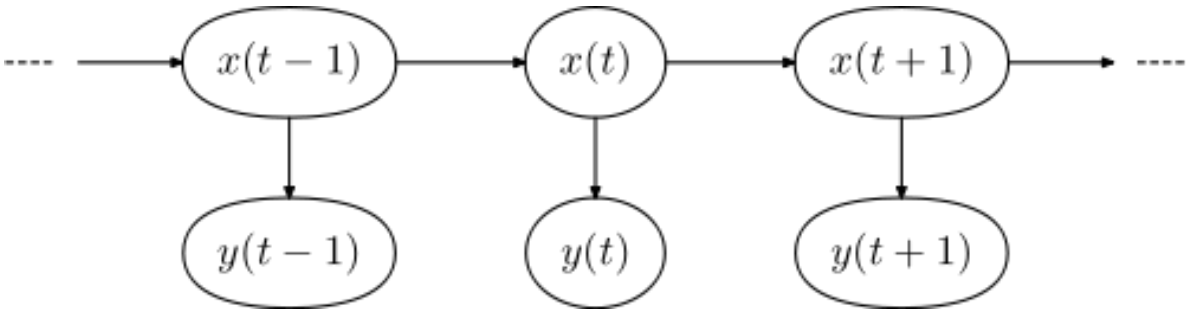
- **Underlying theory**: representative of our beliefs about how markets work.
- **Empirical support**: when applied on real data, results are consistent with empirical evidence.
- **Statistical properties**: captures non-linearities in a simple, parsimonious, and tractable way.
- **Noise reduction**: by discretization.
- **Computational complexity**: reduce dataset size.

Start simple

Break depth imbalance into two groups based on the sign.



Hidden Markov Models (HMM)



Observation model: $p(\mathbf{y}_t|z_t)$, where $\mathbf{y}_t$ are the observations, emissions or output.

- What we observe...

Homogeneous state model: discrete-time, discrete-state first-order Markov chain $z_t \in \{1, \ldots, K\}$ driven by $p(z_t|z_{t-1})$, where K is the number of latent states.

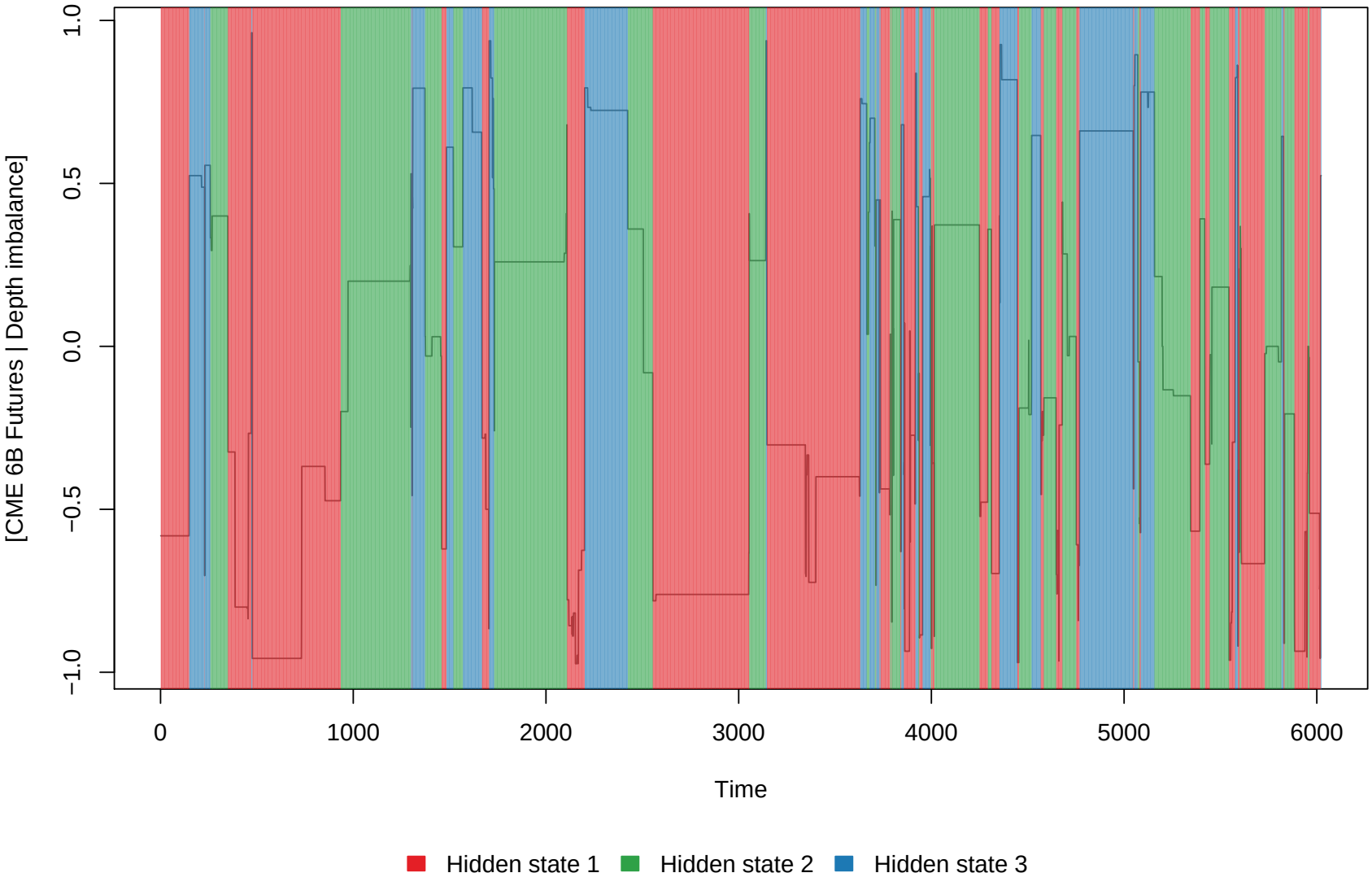
- What is hidden but we would like to know...

Generative model

Another way to see this process:

1. Generate **parameters** according to the priors $\theta^{(0)} \sim p(\theta)$.
2. Generate the **hidden path** $\mathbf{z}_{1:T}^{(0)}$ according to the transition model parameters.
3. Generate the **observed quantities** based on the sampling distribution $\mathbf{y}_t^{(0)} \sim p(\mathbf{y}_t | \mathbf{z}_{1:T}^{(0)}, \theta^{(0)})$.

An example


[CME 6B Futures | Depth imbalance]
1.0
0.5
0.0
−0.5
−1.0
0
1000
2000
3000
4000
5000
6000
Time
Hidden state 1
Hidden state 2
Hidden state 3

Why HMM?

There are other alternatives for time series, but . . .

- **Non-linear**: clustering, bursts, sudden changes.
- **Time aware**: compare with mixture of densities.
- **Markovian memory**: parsimony & tractability.
- **Online learning**: forward filter.
- **Correlation** over long periods.
- And **highly flexible**. . .

Other keywords: sudden jumps, non-linear breaks, regime switching.

How flexible is “flexible”?

Spoiler: **very**!

We may let the behavior of the observations change drastically across states:

- **Location**: low versus high values.
- **Spread**: low versus high volatility.
- **Densities**: Gaussian at times, Student at others.
- **Even models**!
  - Stationary versus random walk with drift.
  - Autocorrelation versus iid.

Are HMM blackbox-y?

Absolutely not!

- States are hidden, yet **meaningful**.
- They are subject to rich **domain-specific interpretation**.
- The model can be **informed via priors**.
- Mathematically **tractable**.
- Plethora of diagnostics.

A tail of two models

We have two models now, which is our **main** model?

Alternatively, what is more important for us? **Signal or state**?

Practical and epistemological repercussions:

- **Focus on observation**: prediction by averaging across the latent states.
  - E.g. predicting default probability.
- **Focus on state**: classification problem (known as segmentation in ML).
  - E.g. identifying whether risk is on or off.

Rich interpretation

The transition matrix can help us **better understand** our signal.

- **Stationary distribution**.
- **Expected stay time**: in a given state.
- **Hitting time**: until the chain arrives in a given state.
- **Mean recurrence time**: on a finite time span.
- **Expected number of visits**: on a finite time span.

Inference

We can **estimate the state probability** at a given time step.

- **Hard classification**:
  - Assign to the highest probability state, or define cut-off points.
  - Would you execute the same trade under different levels of uncertainty?
  - We can do better.
- **Soft classification**:
  - Use the estimate of state probability as a *slider*.
  - Use the estimate of state probability variance to adjust for uncertainty.

Posterior predictive check

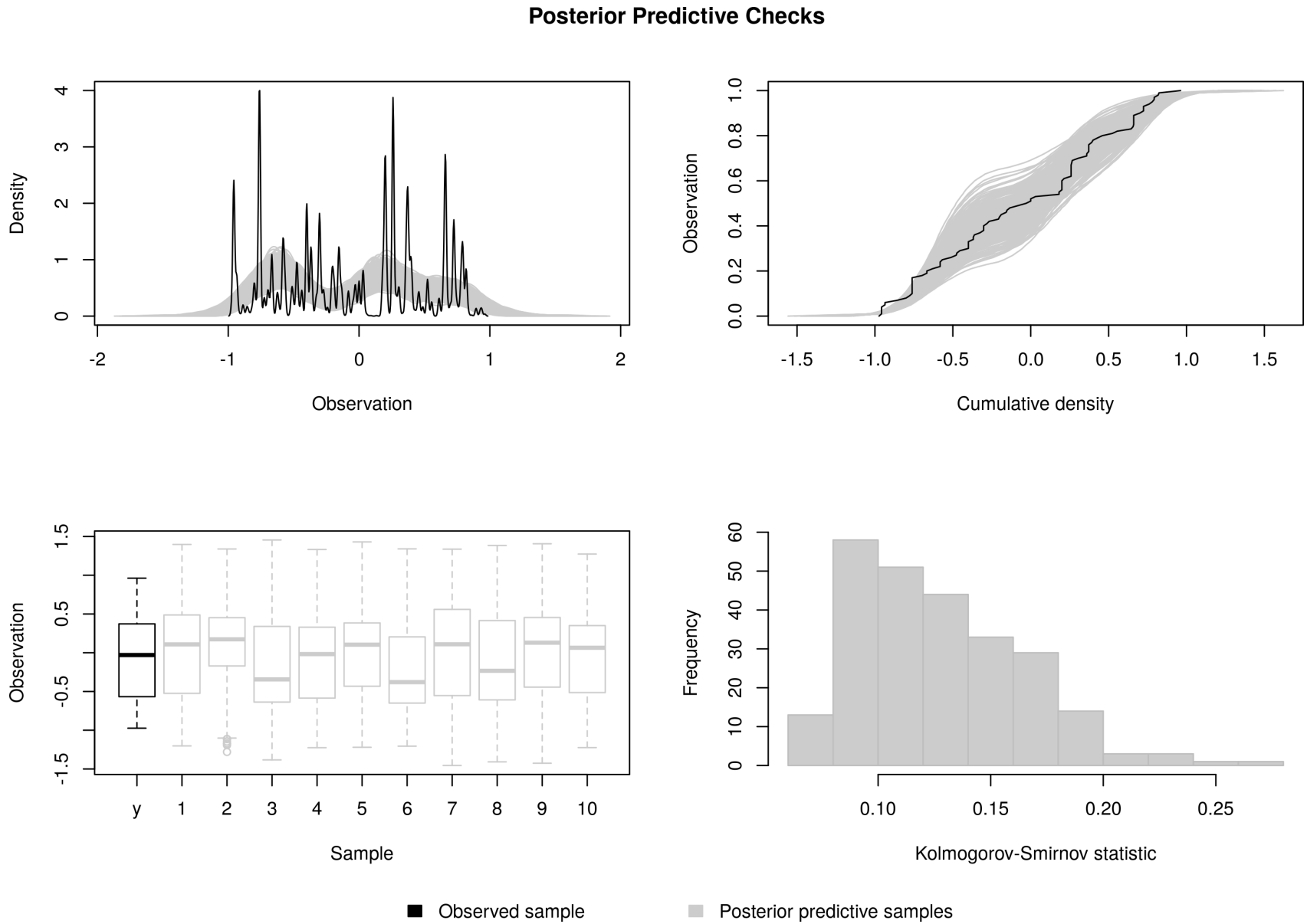
> *If the model fits, replicated data generated under the model should look similar to observed data. To put it another way, the observed data should look plausible under the posterior predictive distribution (Gelman et al. 2013).*

It's an opportunity to test our model on characteristics of the data not directly addressed by the model.

- E.g. ranks, correlations, relationships with explanatory variables.

Posterior predictive check

Can my model generate data similar to the one I observed?



BayesHMM

```
devtools::install_github("luisdamiano/BayesHMM")
```

BayesHMM

- **Powerful**: Full Bayesian inference built on by Stan.
- **Input interface**: intuitive, expressive, friendly to non-statisticians.
- Designed with **statistical carefulness** in mind.
  - One-stop print function: model description, Monte Carlo posterior estimates, MCMC convergence diagnostics, reproducibility notes.
  - Built-in Bayesian validation protocol (Cook 2006, Talts 2018).
  - Built-in posterior predictive checks (Gabry 2019).
- Highly **flexible**:
  - More than 20 densities functions.
  - Time homogeneou and heterogenous transition probabilities.
  - Current limitations: fixed effects, multiple subjects.

References

Carpenter, Bob, Andrew Gelman, Matthew D. Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. 2017. "Stan: A Probabilistic Programming Language." *Journal of Statistical Software* 76 (1). Foundation for Open Access Statistic.

Cook, Samantha R, Andrew Gelman, and Donald B Rubin. 2006. "Validation of Software for Bayesian Models Using Posterior Quantiles." *Journal of Computational and Graphical Statistics* 15 (3). Taylor & Francis: 675–92.

Gabry, Jonah, Daniel Simpson, Aki Vehtari, Michael Betancourt, and Andrew Gelman (2019). "Visualization in Bayesian workflow". Journal of the Royal Statistical Society, Series A: Statistics in Society 182.2, pp. 389–402.

Gelman, Andrew, John B. Carlin, Hal S. Stern, David B. Dunson, Aki Vehtari, and Donald B. Rubin. 2013. *Bayesian Data Analysis, Third Edition (Chapman & Hall/Crc Texts in Statistical Science)*. Chapman; Hall/CRC.

Talts, Sean, Michael Betancourt, Daniel Simpson, Aki Vehtari, and Andrew Gelman (2018). "Validating Bayesian Inference Algorithms with Simulation-Based Calibration". ArXiv 1804.06788. url: https://arxiv.org/abs/1804.06788.

Team, Stan Development. 2017. *Stan Modeling Language: User's Guide and Reference Manual*. Version 2.17.0.